# Towards Secure End-to-End Data Aggregation in AMI through Delayed-Integrity-Verification

Sye Loong Keoh
School of Computing Science, University of Glasgow
9, Woodlands Ave 9, Singapore 738964
Email: SyeLoong.Keoh@glasgow.ac.uk

Zhaohui Tang
School of Infocomm, Republic Polytechnic
9, Woodlands Ave 9, Singapore 738964
Email: Linda_Tang@rp.edu.sg

***Abstract*—The integrity and authenticity of the energy usage data in Advanced Metering Infrastructure (AMI) is crucial to ensure the correct energy load to facilitate generation, distribution and customer billing. Any malicious tampering to the data must be detected immediately. This paper introduces secure end-to-end data aggregation for AMI, a security protocol that allows the concentrators to securely aggregate the data collected from the smart meters, while enabling the utility back-end that receives the aggregated data to verify the integrity and data originality. Compromise of concentrators can be detected. The aggregated data is protected using Chameleon Signatures and then forwarded to the utility back-end for verification, accounting, and analysis. Using the Trapdoor Chameleon Hash Function, the smart meters can periodically send an evidence to the utility back-end, by computing an alternative message and a random value ($m'$, $r$) such that $m'$ consists of all previous energy usage measurements of the smart meter in a specified period of time. By verifying that the Chameleon Hash Value of ($m'$, $r$) and that the energy usage matches those aggregated by the concentrators, the utility back-end is convinced of the integrity and authenticity of the data from the smart meters. Any data anomaly between smart meters and concentrators can be detected, thus indicating potential compromise of concentrators.**

## I. Introduction

With the increasing number of devices being deployed nowadays, ensuring the authenticity and integrity of the information source has become very challenging [11], [13], [18]. Deploying smart meters with two-way communications capability enables energy usage data to be read more frequently e.g., every thirty minutes, and it also allows for the utility companies to push energy prices to the consumer dynamically. Consumers can therefore monitor their energy usage, and plan their energy use according to the price plan in order to save cost. For the utility companies, they can better manage the energy load in terms of power generation, distribution, billing and outage detection through analysis of energy usage data collected through the AMI.

Smart meters are deployed into each household, to measure energy use of electrical appliances that form a Home Area Network (HAN). A Neighborhood Area Network (NAN) [2] is a wireless mesh network that interconnects a group of smart meters with a concentrator, so that the collected energy usage data can then be aggregated before they are forwarded to the Utility Data Management Centre through a WAN interface. In some deployment, the smart meters may have a communication channel with the utility server via 3G or GPRS without needing the concentrator. However, such a communication topology is costly though data aggregation are not required in this case.

Energy data aggregation poses a challenging security risk [12] as the concentrator will usually have access to the energy data obtained from the smart meters, before it performs data aggregation. This means that the concentrator is a single point of failure, and if it is compromised, the attackers will be able to manipulate and tamper with the energy data. In fact, providing end-to-end security between the two end-points, i.e., between the smart meters and the utility data centre, is costly mainly because this can only be provided at the transport or application layer and both end-points are required to establish shared cryptographic keying materials to protect the communication channel end-to-end. The data integrity and the authenticity of the source are not only important for analysis, they are also useful for intrusion detection system.

In June 2010, a powerful worm named *Stuxnet* [17], [1] was uncovered and was known to be able to cripple numerous critical infrastructures including many industrial control systems (ICS). *Stuxnet* exploited the vulnerability of an ICS to compromise the logic controllers. After taking control of the field devices, the compromised logic controller continues to fake "normal" field device readings to be sent to the central controller, thus ensuring that the device compromise remains undetected. As field devices do not protect its messages end-to-end, but rely on the logic controller to aggregate the readings, tampering of data can be conducted easily. If end-to-end security were provided, the logic controller would not have access to the data, and at the same time, the speed of detecting device compromise would be much faster and recovery actions could be triggered earlier. Similar attacks could be launched against AMI [21] by attacking the concentrators in the network, and such an attack can equally be used to paralyze the smart grid power generation and distribution network.

This paper proposes a scheme to provide secure data aggregation through *delayed-integrity-verification* for any application architecture that is based on wireless meshing such as AMI while preserving the efficiency of a distributed architecture. Devices continue to be organized in a hierarchical manner, consisting of smart meters, reporting energy usage data periodically to the concentrator that serves as data aggregator

before the data are forwarded further to the utility back-end. The smart meters periodically generate a chameleon message hash based on the previously reported usage data, to allow for the utility back-end to verify the data's integrity and authenticity, and to perform check on whether the concentrators have tampered with the energy data. Any tampering to the energy data can be detected by the utility back-end and an alarm can be triggered to act on the misbehaving concentrators.

The paper is organized as follows: Section II provides some background and discusses related work. Section III presents the attacker models and security requirements. Section IV outlines the steps in providing *delayed-integrity-verification* to achieve end-to-end integrity and authenticity verification. Section V presents a security analysis of the proposed scheme and Section VI concludes the paper with future work.

## II. Related Work

This section provides background on Chameleon Signatures [15], [14], and related work on data aggregation.

### A. Chameleon Hashing

Chameleon Hashing was introduced by Brassard and Chaum [6]. It has the same properties as the normal hash function except that it has a trapdoor in built for finding collisions. Without the knowledge of the trapdoor, it works as a collision resistant functions on which a regular signature function can be applied to provide authenticity and integrity to the message. Chameleon Hashing is associated with a pair of public and private keys in which the private key serves as the trapdoor for the hash function. It is collision resistant in that without the knowledge of the trapdoor, it is infeasible to find two inputs when hashed are mapped to the same hash value. The distinct capability of Chameleon Hashing is that it allows the owner of the trapdoor to change the input to the function without changing the output.

### B. Chameleon Signatures

Chameleon Signature [15], [14] was introduced as a much simpler implementation of undeniable signatures [8], [7]. It is built similar to the traditional digital signature scheme that is *hash-then-sign* approach. A regular digital signature scheme such as RSA, DSS or ECC is applied to a special type of hashing called *chameleon hash functions* [6] as described in Section II-A.

The Chameleon Signature scheme allows a Signer, $S$ to sign a message to be sent to a Recipient, $R$ such that it gives $R$ the ability to *forge* further signatures of $S$ at will. Consequently, when presenting the signature to a third party, it is not possible to prove the validity of the signature because $R$ could have produced such a signature by himself. However, this scheme is useful in that it provides the signer $S$ with the *exclusive* ability to prove that a forged signature is in fact a forgery. The signer only needs to provide a short piece of information as evidence for the forgery, i.e., the original message in which when hashed produces the equivalent hash value as claimed in the forged signature.

Although this scheme is conceptually simpler and more efficient in that it is non-interactive, it appears that its application is somewhat restricted. In this paper, we adapted the use of Chameleon Signatures to enable secure data aggregation in a typical wireless mesh network where data produced by multiple sources are aggregated by a third entity before they are forwarded to the final Recipient for analysis.

### C. Sanitizable Signatures and Transitive Signatures

Sanitizable signature [3] was introduced to allow a signer to partly delegate signing rights to a semi-trusted party called a sanitizer, so that it is allowed to change a pre-determined part of the signed document. This is particularly useful for applications such as authenticated multicast, and authenticated database outsourcing because multicast messages can be customized by a trusted sanitizer without compromising the integrity and authenticity of the messages from the source. However, this scheme has a strong assumption that the sanitizer is semi-trusted, and it is difficult to detect dishonest sanitization in this scheme.

Transitive signatures [19], [4] and aggregate signatures [5] provide a mechanism to transform and aggregate multiple signatures into one respectively. Thus allowing for the verification of the authenticity and integrity of the messages from its original sources. With aggregate signature, $n$ signatures on $n$ distinct messages from $n$ distinct users can be aggregated into a single short signature. This single signature (and the $n$ original messages) will convince the verifier that the $n$ users did indeed sign the $n$ original messages [5]. However, using such schemes would incur additional communication overhead as each source must compute a digital signature for each message it transmits before they are aggregated.

### D. Privacy-Preserved Data Aggregation

Danezis et. al. [9] proposed an aggregation scheme based on secret-sharing and secure multi-party computation techniques. Meter readings are jointly processed by a public storage service and other independent authorities, each owing an additive share of the readings. Other privacy-friendly data aggregation schemes include [16], [10], [20], they are efficient in that the meters generate readings that are blinded by additive shares summing to zero. When the blinded readings are revealed and summed, the sum of the readings is obtained.

However, most of the privacy-preserved data aggregation scheme assumes that the smart meters and the concentrators are trustworthy. In this paper, we emphasize on the authenticity and integrity of the smart meter readings, ensuring that no one has tampered with the readings during the transmission. Therefore, guaranteeing the provenance and data integrity can be seen as a step prior to privacy preservation of the meter readings.

## III. Security Threats and Requirements

### A. Attacker Model

This section describes three main security attacks on an AMI. In this paper, we do not consider physical jamming attacks and distributed denial-of-service attacks.

*1) Eavesdropping on Communication Channel:* Attackers can eavesdrop on the communication channel between the smart meter and the concentrator, as well as the channel between the concentrator and the utility back-end to obtain meter readings, commands and aggregated energy usage data.

*2) Man-in-the-Middle Attack:* The concentrator is regarded as the MITM as it sits between the smart meter and the utility back-end. Attackers can compromise the concentrator in order to tamper with the data, and selectively report data to the utility back-end. In a more generalized form, the communication is vulnerable to MITM attacks when attackers are sitting in between any of the AMI entities.

*3) Compromise of Concentrator:* This results in the ability of the concentrators to fully manipulate the smart meters, e.g., remote disconnect the electricity and reports fraudulent data to the utility back-end to enable theft of electricity. Attackers are motivated to hack the concentrators, e.g., to tamper with the aggregated data and manipulate the energy data of many households.

### B. Security Requirements

Based on the smart grid and AMI communication architecture, we derive three main security requirements as follows:

*1) Data Integrity:* The energy data used by each household must be integrity protected to prevent energy theft. Any tampering of the data during transmission to the utility back-end would not be acceptable. The accuracy of the energy data must reflect the current energy load, and it is important to balance the energy demand and response.

*2) Data Origin Authentication:* The data origin of the measurement is important to ensure that it was taken using a designated device. This enables the utility back-end to securely map the energy usage data to a smart meter. Such a guarantee is crucial to enable the utility to remotely determine the status of smart meters and concentrators, e.g., performing fault diagnostic to identify the faulty or misbehaving concentrators and smart meters in the AMI.

*3) Secure Data Aggregation:* As most of the data are being aggregated by the concentrators, this means that the original data have been transformed. Although this poses difficulty in ensuring the data integrity and data originality, a very important security requirement is to ensure that the utility back-end that receives the transformed or aggregated data must have the ability to check the integrity and source authenticity. In addition, an AMI should be able to detect any unauthorized data modification by the concentrators, application hosting devices, or any other intermediaries, so that any incidents of intrusion can be detected and acted upon swiftly.

## IV. Secure E2E Data Aggregation

This section introduces an approach to end-to-end data aggregation for AMI. It allows the concentrators to securely aggregate data collected from smart meters, while enabling the utility back-end that receives the aggregated data to verify the data origin and its integrity.

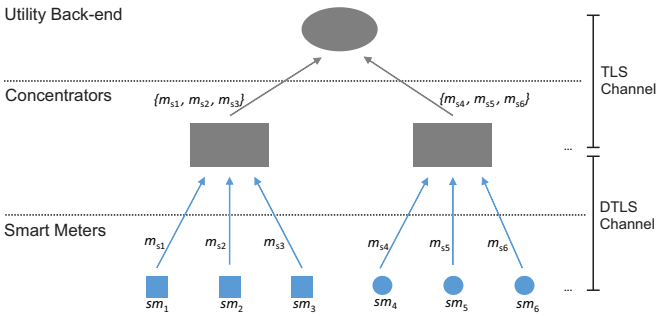


Fig. 1. Data flow in an Advanced Metering Infrastructure (AMI)

### A. Data Flow in AMI

Fig. 1 illustrates the data flow in AMI. Smart meters periodically send energy usage data to the concentrator. Energy usage data from multiple sources are aggregated by the concentrator in order to minimize bandwidth consumption and reduce the number of messages to be transmitted, therefore improve efficiency. The aggregated data are forwarded to the utility back-end for analysis. Typically, two secure communication channels are needed to protect the authenticity and integrity of the data. Between the concentrators and the utility back-end, digital signature is used to provide authenticity and data integrity. If encryption is needed, a TLS channel can be established between the concentrators and the utility back-end. It is possible that there are additional intermediaries between the concentrators and the utility back-end such as routers, gateways, etc. However, their roles are restricted to routing and forwarding of the protected data to the central controller. Between the smart meters and the concentrators, a symmetric-key communication channel can be used to protect information between the two parties.

### B. Setup, Operation and Verification

The proposed solution is based on Chameleon Hashing [6] and Chameleon Signatures [15], [14] in that it allows for the concentrators to aggregate the data, compute a chameleon hash value ($CHV$) of the aggregated data, and then sign it using a traditional digital signature. The aggregated data together with the signature are sent to the utility back-end, thus establishing the authenticity of the data from the concentrators. The smart meters which know the trapdoor function of the Chameleon Hash Function can *periodically* compute a different message, $m_i'$ and a random value $r_i$, given the $CHV_{cc}$ computed by the concentrator, where $m_i'$ consists of all the previous energy usage data logged by the smart meter, $sm_i$ within a time period. The combination of ($m_i'$, $r_i$) are then forwarded to the utility back-end for verification. It serves as an evident to prove that the recorded energy usage data are truly originated from the smart meter itself. This data could either be routed via the concentrators or via an alternative communication channel if it exists, e.g., a GRPS/3G connectivity between the smart meter and the utility back-end. The utility back-end is then convinced of the integrity of the energy usage data and its data origin because ($m_i'$, $r_i$) when hashed, the resulting chameleon hash

value is equivalent to the $CHV$ sent by the concentrators, and that the recorded energy usage values match the corresponding values previously sent by the concentrators. Such a property proves that the concentrators have not been compromised.

### C. Commissioning

Prior to deployment, all smart meters and concentrators must be commissioned with the relevant cryptographic keying materials. As illustrated in Fig. 2, when the AMI is first deployed with smart meters $sm$, the meters are grouped together and then bound to a concentrator. Each group of smart meters is then commissioned with a *Trapdoor Chameleon Hash Function*, where they share the trapdoor key, $K'$ among themselves, while the responsible concentrator is configured with the corresponding *Chameleon Hash Function* which has the public key, $K$. This public-key held by the concentrator is also known to the utility back-end. Such a configuration allows both the concentrator and the utility back-end to compute a Chameleon Hash $CHV$ given a message, while only the smart meters have the ability to produce the same $CHV$ with a different message. Apart from the smart meters in the group, no other entity in the system should know the *Trapdoor Chameleon Hash Function* and its trapdoor key $K'$.

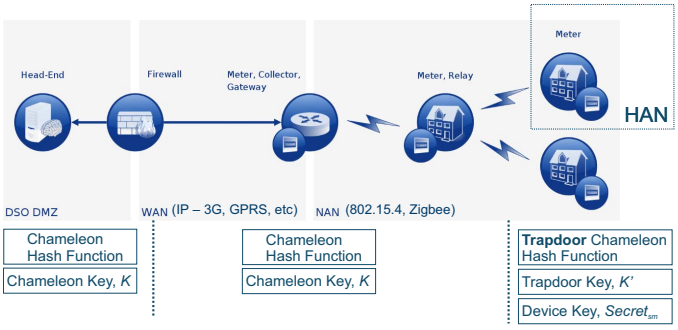


Fig. 2. Commissioning of cryptographic keying materials

Each smart meter is also provisioned with a unique symmetric key, $Secret_{sm}$ for identification purposes. This is done as part of the manufacturing process, and the utility back-end has knowledge of this key so that secure communication can be established to facilitate firmware update, dissemination of energy tariffs, updating of device configuration, etc.

It is also assumed that there is a secure channel between the concentrator with each smart meter it manages. A symmetric-key encryption scheme, e.g., DTLS is used to protect the confidentiality and integrity of the data transmitted. This is essential as the wireless communication is vulnerable to passive eavesdropping, packet injection and tampering.

The concentrator typically has more computational capability and resources, therefore a digital signature scheme can be deploy to protect the aggregated data to be sent to the utility back-end. With this, each concentrator is also commissioned with a public-private keypair, and it can be used to sign the aggregated data using its private key. The utility back-end has knowledge of all the public-keys of all concentrators under its jurisdiction, so that signatures can be verified.

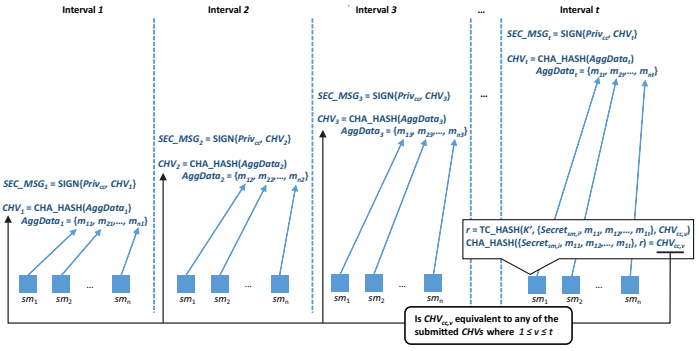


Fig. 3. Overview of secure E2E data aggregation with *delayed-integrity-verification*

### D. Operations

Fig. 3 shows the cryptographic operations to be performed by the concentrators and the smart meters to ensure the integrity and authenticity of the data. The details of each operation are described in the following sections.

*1) Transmission of Energy Usage Data:* For a group of smart meters $sm_1$, $sm_2$, $sm_3$,..., $sm_n$ where $n$ is the number of meters bound to a concentrator, $cc$, each smart meter periodically reports energy usage data to the concentrator for aggregation. The usage data is encrypted with each smart meter's respective secret-key shared with the concentrator. For example, $sm_i$ sends $m_{i,j}$ to the concentrator, where $i$ is the device identifier, and $j$ is the message number. This message can be protected using DTLS Record Layer, thus providing message freshness guarantee and detecting message replay.

*Operation:* **Transmit Data**
$sm_i \rightarrow cc$: $\{sm_i, m_{i,j}\}$

*2) Data Aggregation:* When the concentrator, $cc$ receives data from multiple smart meters it manages, it is responsible for aggregating the data received before forwarding them to the utility back-end. The concentrator collects $n$ messages from the smart meters periodically, i.e., $m_{1,j}$, $m_{2,j}$,..., $m_{n,j}$ where $n$ is the number of smart meters, and $j$ is the message number.

*Operation:* **Aggregate Data**
$AggData_k$: $\{m_{1,j}, m_{2,j},..., m_{n,j}\}$
where $k$ is the aggregated message identifier and $k > 0$

The aggregated data is hashed and then signed using the concentrator's Chameleon Hash Function and private key respectively.

*Operation:* **Hash and Sign Aggregated Data**
$CHV_{cc,k}$ = CHA_HASH($K_{cc}$, $\{AggData_k\}$)
$SEC\_MSG_{cc,k}$ = SIGN($Priv_{cc}$, $CHV_{cc,k}$)
where CHA_HASH is a Chameleon Hash Function.

Finally, the aggregated data, and the signature are sent to the utility for verification. At the same time, the Chameleon Hash Value, i.e., $CHV_{cc,k}$ is broadcast to all the smart meters managed by the concentrator as a means to acknowledge receipt of the data transmitted by the smart meters.

*Operation:* **Transmit to Utility**
$cc \rightarrow$ *utility*: $SEC\_MSG_{cc,k}$, $\{AggData_k\}$
$cc \rightarrow sm_{1,2,...,n}$: $CHV_{cc,k}$

*3) Verification:* Upon receipt of the aggregated data and signature from the concentrator, the utility back-end verifies the digital signature using the concentrator's public-key. During the signature verification process, the utility back-end uses the concentrator's Chameleon Hash Function to compute the $CHV$. If the verification is successful, it accepts the integrity and authenticity of the data received, and stores the $CHV$ and the $AggData$ for *end-to-end* verification later on.

*Operation:* **Verify Signature**
VERIFY($Pub_{cc}$, $AggData_k$, $SEC\_MSG_{cc,k}$)

### *E. Periodic End-to-End Verification*

Since the signature is generated by the concentrators, the utility back-end can only believe that the data are originated from the concentrators and have not been tampered with during transmission. However, there is no guarantee that the concentrators have not been compromised and tampered with the readings before aggregating them. Therefore, in order to ensure that the concentrators have not been attacked, and the reported data are truly originated from the smart meters (with the assumption that the smart meters are trusted[1]), the *delayed-integrity-verification* scheme as described in this section must be enabled.

*1) Transmission of Evidence:* The *delayed-integrity-verification* scheme divides transmission time into multiple fixed length period, where each period consists of $t$ intervals. This allows the smart meters to send up to $t$ messages to the concentrator in a period. As mentioned previously, for each message that the smart meter had sent, it receives an Acknowledgement that contain a $CHV_{cc,i}$ from the concentrator, where $i$ is the interval and $cc$ denotes the identifier of the concentrator. At the end of each time period, each smart meter uses any of the received Chameleon Hash Value within the period, e.g., $CHV_{cc,v}$ where $1 \leq v \leq t$ and the corresponding *Trapdoor Chameleon Hash Function* to compute a value, $r_i$ such that the concatenation of its device key with all the respective data they had sent for that period, $\{Secret_{sm,i}, m_{i,1}, m_{i,2},..., m_{i,t}\}$ when hashed together with $r_i$ is equivalent $CHV_{cc,v}$. All smart meters that are bound to the same concentrator use the same *Trapdoor Chameleon Hash Function*.

[1]Collusion between smart meters and concentrators are beyond the scope of this paper. This is because if the information source, i.e., the smart meter is dishonest, it is extremely hard to guarantee the authenticity and originality of the energy usage data.

*Operation:* **Each smart meter,** $sm_i$ **computes** $r_i$
$r_i$ = TC_HASH($K'$, $\{\{Secret_{sm,i}, m_{i,1},..., m_{i,t}\}, CHV_{cc,v}\}$)
where
$CHV_{cc,v}$ = CHA_HASH($K$, $\{Secret_{sm,i}, m_{i,1},..., m_{i,t}\}$, $r_i$)

Since only the smart meters know the *Trapdoor Chameleon Hash Function*, no other entity can produce a different ($m'$, $r'$) when hashed, is equivalent to $CHV_{cc,v}$. All the ($m_i$, $r_i$) are then forwarded to the utility back-end. The Smart Meter's unique device key, $Secret_{sm,i}$ is also concatenated with the energy data for identification purpose. The smart meter is only required to perform the *Trapdoor Chameleon Hash* operation, without needing to generate any signatures. The reported ($m_i$, $r_i$) are used as evidences to detect any malicious actions performed by the concentrator. The $Secret_{sm,i}$ ensures that no one can impersonate the smart meter, while the ($m_i$, $r_i$) if corresponds to the energy data previously reported by the concentrators ensures data consistency, and that the concentrators have not been compromised. Even though the evidence is routed via the concentrators, if it was dropped, the utility back-end would be able to detect this easily and suspect misbehaviour of the concentrator.

*Operation:* **Report to Utility**
$sm_i \rightarrow utility$: $sm_i$, ($\{m_{i,1},..., m_{i,t}\}$, $r_i$)

*2) Delayed-Integrity-Verification:* The utility back-end stores all the data received from the concentrators, including the aggregated data ($AggData$), as well as all the Chameleon Hash Values (CHVs). When it receives ($m_i$, $r_i$) from each smart meter, it can verify the integrity and authenticity of the messages by computing a Chameleon Hash Value of ($m_i$, $r_i$), $CHV_{verify}$. If the $CHK_{verify}$ is equivalent to any of the received $CHV_{cc,v}$ from the concentrator managing the smart meters, then it also accepts the signature.

*Operation:* **Verify end-to-end security**
CHV_VERIFY($K_{cc}$, $\{Secret_{sm,i}, m_{i,1},..., m_{i,t}\}$, $r_i$)

The utility back-end also checks each individual message whether it matches the data previously signed by the concentrator. A match in terms of message values, and the number of reported data indicates that both the concentrator and the smart meter are in agreement and that the data have not been tampered with. The utility back-end believes that the data originated from the smart meters, thus achieving truly end-to-end security between the smart meters and the utility back-end. If there's an anomaly in terms of message values, the concentrator is suspected to be faulty or compromised. In addition, a mismatch in terms of number of reported data implies that the concentrator selectively dropped the data (with the assumption that the channel between the concentrator and the utility back-end is reliable, and the smart meters are assumed to be trusted). Consequently, attacks on the concentrators can be detected swiftly through this scheme.

## V. Security Analysis

### A. Data Integrity

The integrity of energy usage data is guaranteed by the secure channels in the AMI. Assuming that the DTLS channel between the smart meter and the concentrator is secure, it is sufficient that the integrity of the data transmitted in the channel is securely protected. Likewise, a reliable secure channel, i.e., TLS between the concentrators and the utility back-end guarantees that the aggregated data is integrity protected. This TLS channel is reliable in that all the data transmitted is received by the utility back-end, so that billing can be performed correctly.

### B. Data Origin Authentication

The device key, $Secret_{sm_i}$ as shown in Fig. 3 guarantees that the data is originated from the smart meter $sm_i$, since $sm_i$ is the only entity which shares the device (secret) key with the utility back-end. When the smart meter performs *delayed-integrity-verification* protocol by sending $(m_i', r_i)$ to the utility back-end, this message is concatenated with $Secret_{sm,i}$ which can only be produced by the smart meter itself.

### C. Secure Data Aggregation

Assuming our chameleon digital signature is built by applying a regular unforgeable digital signature such as RSA (or DSS, ECC) to a discrete logarithm based hashing (Fig. 1 in [15]). We claim that the proposed *delayed-integrity-verification* scheme achieves end-to-end data integrity.

It suffices to prove that unforgeability of the proposed scheme: no third party can provide a different $(m', r')$ when hashed, is equivalent to $CHV_{cc,v}$. In order to forge such a different $(m', r')$, a third party needs to break the underlying digital signature scheme, or to find collision in the chameleon hashing. Breaking the underlying digital signature scheme contradicts with the unforgeability of the underlying digital signature, while finding collision in the chameleon hashing implies computing the secret *Trapdoor Chameleon Hash Function* and thus contradicts with the definition of chameleon hashing. Henceforth, the proposed scheme achieves end-to-end data integrity.

## VI. Conclusions

We have provided a novel use of Chameleon Signatures other than its traditional usage, to detect misbehavior of concentrators in AMI. Using the *delayed-integrity-verification* scheme, we can ensure that the data aggregated by the concentrator is protected end-to-end in that the integrity, authenticity and data originality of the energy usage data can be guaranteed. This would be beneficial to the protection of critical infrastructures. The advantage of this scheme is that the digital signatures generated by the concentrators can be used to verify both the data reported by the smart meters and the concentrators themselves. The smart meters are not required to use any public-key based signature scheme, but merely executing a (Chameleon) hash operation.

We have also provided a security analysis to first show the feasibility and robustness of the *delayed-integrity-verification* scheme. The natural next step is to simulate and implement it in an AMI test bed to assess the performance of the protocol.

Another future direction is to integrate the protocol with an intrusion detection system. Machine learning techniques can be used to detect any anomaly in the meter readings.

## References


[1] Kaspersky Lab provides its insights on Stuxnet worm, Sept 2010.

[2] Wi-SUN Alliance, URL: http://www.wi-sun.org/, 2014.

[3] G. Ateniese, D. H. Chou, B. de Medeiros, and G. Tsudik. Sanitizable signatures. In *Proceedings of the 10th European Conference on Research in Computer Security*, ESORICS'05, pages 159–177, Berlin, Heidelberg, 2005. Springer-Verlag.

[4] M. Bellare and G. Neven. Transitive signatures: new schemes and proofs. *Information Theory, IEEE Transactions on*, 51(6):2133–2151, 2005.

[5] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Proceedings of the 22Nd International Conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT'03, pages 416–432, Berlin, Heidelberg, 2003. Springer-Verlag.

[6] G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, Oct. 1988.

[7] D. Chaum. Zero-knowledge undeniable signatures. In *Advances in Cryptology - EUROCRYPT '90, Workshop on the Theory and Application of of Cryptographic Techniques, Aarhus, Denmark, May 21-24, 1990, Proceedings*, volume 473 of *Lecture Notes in Computer Science*, pages 458–464. Springer, 1990.

[8] D. Chaum and H. Antwerpen. Undeniable signatures. In G. Brassard, editor, *Advances in Cryptology (CRYPTO'89) Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 212–216. Springer New York, 1990.

[9] G. Danezis, C. Fournet, M. Kohlweiss, and S. Zanella-Béguelin. Smart meter aggregation via secret-sharing. In *2013 Smart Energy Grid Security Workshop, SEGS 2013*. ACM, 2013.

[10] B. Defend and K. Kursawe. Implementation of privacy-friendly aggregation for smart grid. In *2013 Smart Energy Grid Security Workshop, SEGS 2013*. ACM, 2013.

[11] Z. Fan, P. Kulkarni, S. Gormus, C. Efthymiou, G. Kalogridis, M. Sooriyabandara, Z. Zhu, S. Lambotharan, and W. H. Chin. Smart grid communications: Overview of research challenges, solutions, and standardization activities. *Communications Surveys Tutorials, IEEE*, 15(1):21–38, 2013.

[12] T. B. Florian Skopik, Zhendong Ma and H. Grneis. A survey on threats and vulnerabilities in smart metering infrastructures. *International Journal of Smart Grid and Clean Energy*, September 2012.

[13] N. Khurana, M. Hadley, N. L., and D. Frincke. Smart-grid security issues. *Security and Privacy, IEEE*, 8(1):81–85, 2010.

[14] H. Krawczyk and T. Rabin. Chameleon hashing and signatures, 1997.

[15] H. Krawczyk and T. Rabin. Chameleon signatures. In *NDSS*. The Internet Society, 2000.

[16] K. Kursawe, G. Danezis, and M. Kohlweiss. Privacy-friendly aggregation for the smart-grid. In *Privacy Enhancing Technologies - 11th International Symposium, PETS 2011, Waterloo, ON, Canada, July 27-29, 2011. Proceedings*, pages 175–191. Springer, 2011.

[17] R. Langner. To Kill a Centrifuge: A Technical Analysis of What Stuxnet's Creators Tried to Achieve. Technical report, The Langner Group, Nov 2013.

[18] F. Li and B. Luo. Preserving data integrity for smart grid data aggregation. In *Smart Grid Communications (SmartGridComm), 2012 IEEE Third International Conference on*, pages 366–371, Nov 2012.

[19] S. Micali and R. Rivest. Transitive signature schemes. In B. Preneel, editor, *Topics in Cryptology CT-RSA 2002*, volume 2271 of *Lecture Notes in Computer Science*, pages 236–243. Springer Berlin Heidelberg, 2002.

[20] E. Shi, R. Chow, T. h. Hubert Chan, D. Song, and E. Rieffel. Privacy-preserving aggregation of time-series data. In *Network and Distributed System Security Symposium (NDSS)*, 2011.

[21] G. N. Sorebo and M. C. Echols. *Smart Grid Security: An End-to-End View of Security in the New Electrical Grid*. CRC Press, February 2012.